# **F28HS – Coursework 2 Report**

Specification: In this project, the group designed the well-known ‘Mastermind’ boardgame in C and ARM assembler /blah /blah

Table of Contents

[F28HS – Coursework 2 Report 1](#_Toc99450454)

[Hardware Specification: 2](#_Toc99450455)

[Code Structure & Functionality of main 3](#_Toc99450456)

[Structure of Main Function 4](#_Toc99450457)

# Hardware Specification:

For the configuration which can be seen in the image below, a button and two LED setup was utilised. The button allowed acted as an interface which consequently allowed users to interact with the program. The two LED’s, on the other hand were outputs for the program which gave the user feedback on their input.

A picture containing text, indoor, businesscard

Description automatically generated

BCM PIN Numbering:

* Green LED was connected to BCM PIN 13, or physical pin 33.
* Red LED was connected to BCM PIN 5, or physical pin 29.
* Button was connected to BCM PIN 19, or physical pin 35.

# Code Structure & Functionality of main

* Delay & Delay Microseconds – These two functions allowed us to implement ‘waits’ in our program. This would stop the flow of execution for a specified amount of time, which was useful for displaying outputs etc.
* digitalWrite – This function allowed us to set and clear pins in the program. If a ‘1’ was passed, it would set a pin, or if a ‘0’ was passed it would clear a pin. This was useful for turning on/off LED’s.
* pinMode – This function changes the mode of a particular pin to either an input or output. This is used when the game is being setup, by changing the button to an input, and the two LED’s to outputs.
* readButton – This function determines whether the button is currently pressed or not at the time of invocation. It will return 1 if it is, or 0 otherwise. There is also an extra ‘buttonDown’ condition on the function. This guard will only register a button press is buttonDown = 0, meaning the button is ‘up’. This prevents multiple registers of a button press.
* waitForButton – This is the function that is used in main while loop. This invokes the previously mentioned ‘readButton’ and implements a delay. There is an extra guard on this function also. This guard prevents the button registering greater than 3 button presses for a single colour.
* initSeq – This function is invoked when a secret sequence has not been provided. This initialises a **random** sequence for the pins.
* showSeq – This displays a given sequence to stdout.
* displaySeq – This displays a given sequence using the LED’s. This was used for debugging.
* dCopy – This function allowed us to perform a deep copy of ‘theSeq’ and ‘attSeq’ so that we were not manipulating the original values when it came to countMatches.
* countMatches – Takes two arguments and displays how many colours in the second sequence match the first sequence.
* showMatches – Displays the exact and approximate matches for an input to stdout.
* readSeq – Parses an integer i.e. 133 and puts it into a sequence in the form of ‘1,3,3’. This will then correspond to the colours **red, blue, blue.** This is used when command-line arguments -s or -u are used.
* readNum – Reads a sequence of 3 numbers from stdin and stores the values in an array. Used for testing game logic.
* time\_handler – Callback function that was used for our timer, it changes the ‘waitingFlag’ static variable once a given period of time has elapsed.
* initITimer – This sets up the timer and attaches timer\_handler as its callback function.
* blinkN – This blinks an LED a specified amount of times.
* clearPins – This turns off both LED’s.
* echoMatches – This displays to the user how many exact and approximate matches has been found, through the use of the LED’s blinking.
* displaySuccess – This LED blinking sequence in invoked to let the user know that their guess was correct.

## Structure of Main Function

1. At the start of the main function, the seed of the random function is based upon the system’s clock.
2. Variables are then assigned.
   1. Local variables from the static
3. The code then takes care of the command-line arguments that were passed and dynamically allocates memory to the four sequence arrays (seq1, seq2, cpy1, cpy2, attSeq)
4. The pin mode is then set for the LED’s and the buttons.
5. The outer while loop is now entered, this is the ‘main flow’ of the program. This outer while loop will continue indefinitely until the correct guess has been inputted.
6. The nested while loop then has the condition found < seqlen. This essentially keeps track of how many colours have already been inputted and will break from this while loop once found equals seqlen, to compare and find how many matches were in the guess.
7. When the while loop is entered, the static variable waitingFlag is set to one.
   1. Next, the program will ‘wait’ until the first button press of the colour sequence has been inputted.
   2. Once the first button press has been registered, the timer is initialised.
   3. The program will then enter another while loop with the condition of while ‘waitingFlag’. This while loop will continually wait & count for button presses until the timer is finished, and the callback is fired which changed the static variable ‘waitingFlag’ to 0.
8. LED’s will blink to give feedback to the user about their input, and the timer reset.
9. The program will then compare the user’s colour sequence to the secret sequence.
   1. This will then blink the LED’s of how many exact/approximate value were found respectively.
10. If the user’s guess was correct, the LED’s will blink a specific way indicating to the user that their guess was correct. The game will then terminate. Otherwise, the game will continue to the next guess.

# List of Functions That Directly Access Hardware

pinMode()

digitalWrite()

readButton()

# Name & Interface Discussion

We chose variable and method names that were relevant to their use and function.

# Sample Execution in Debug Mode

Graphical user interface, application

Description automatically generated

# Link to Video on OneDrive:

[mm-demo.MOV](https://heriotwatt-my.sharepoint.com/:v:/g/personal/jd2002_hw_ac_uk/EaKuxU_Iu35LiXvLPeYobdEBF4L4BWBdALqFxigqSNOkLQ?e=ftcOrQ)